# Oscillating activation functions and other recent progress in activation functions: A survey

**Anuran Roy**
anuran.roy2020@vitstudent.ac.in

## Abstract

Neural networks are at the heart of many autonomous intelligent systems now. Originally designed to mimic human neurons forming the brain, modern neural networks have their fundamental component in the perceptron[1][2], an analog to a single neuron in the brain. A perceptron boils down to a set of input, an equal number of weights, biases, and an activation function that maps the inputs to a bounded output. A perceptron can represent the basic logic gates, except the XOR gate, popularly referred to as the XOR problem[3]. Recent work [4] have proposed biologically inspired oscillatory functions that solve the XOR problem with a single perceptron. We highlight the shortcomings of existing activation functions with respect to the XOR problem and how oscillating functions can solve them in this paper.

## 1 Introduction

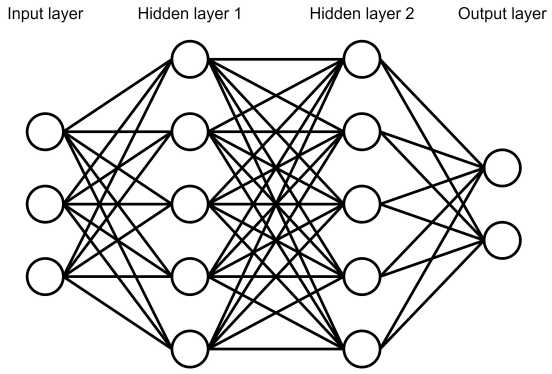


Figure 1: Structure of a Multilayer Neural Network

At their core, neural networks are about fitting non-linear functions through maximum possible sample space, which involves hyperplanes involved with linear separation.

At the core of neural networks is the concept of statistical learning, making use of a set of statistical mechanisms, notable of which are:

- **Cost Function**: The cost function $f$ is defined as the function that gives a measure of the accuracy (or loss) of the predictions given by the neural network. It is the primary component for the Gradient Descent Algorithm (discussed below), and is considered to be an ideal function if it is differentiable everywhere and is computationally inexpensive.
- **Forward Propagation**[5]: For training neural networks with right parameters, intermediate states are highly helpful as they help in breaking down the entire mechanism of gradient descent into

a number of small incremental steps, with intermediate variables that enable for more efficient convergence of the loss values from the loss function. This is referred to as Forward Propagation (or sometimes as Forward Pass).

- **Backward Propagation (Backpropagation)**[6]: Backpropagation is a standard approach used in speeding up the process of training in most neural networks, to fit the neural network. It calculates the derivative (also called gradient) of the loss function with respect to the weights for a single pair of input-output. Using the concept of memoization in dynamic programming, it efficiently computes the changes in the weights required to minimize loss. It works by using the chain rule, computing for one layer at a time. It then iterates backwards in the chain rule, while keeping on updating weights.
- **Gradient Descent**: Gradient descent (also known as GD) is a class of iterative approaches to find a local minimum of the cost function.
  For every weight $w_{ij}$ of a neural network, the gradient descent formula is given by:

$$w_{ij} = w_{ij} - \alpha.\frac{\partial f(S_{i=0}^{n} x_i)}{\partial w_{ij}}$$

  where:
  $\alpha$ is a hyperparameter called the learning rate,
  $f$ is the cost function consisting of parameters.

  The termination condition for the gradient descent is when the value of the derivative becomes zero.

## 2 Background

Every neuron in a neural network needs to map the weighted sum of its input to a single output. The function that maps this is called the **Activation Function**.

Formally, we define an activation function $A$ as:

$$A : Q \to E, \qquad A(w_0 + \sum_{i=1}^{n} w_i \times x_i) \ \ \epsilon \ \ E$$

where:

$Q$ is the set of inputs,
$E$ is the set of possible outputs,
$n$ is the number of input parameters (or weights),
$w_i$ is a weight,
$w_0$ is the bias, and
$x_i$ is a parameter.
Activation functions are of prime importance in neural networks, mapping a wide variety of inputs to a bounded output by taking in weighted inputs.

## 3 Activation function properties

A good activation function must have the following properties:

- **Non-linear**: A good activation function must be non-linear. According to *Hornik et. al,* [7], non-linear functions are universal approximators. Conversely, if an activation function is linear, using it in multiple layers would be equivalent to a single layer model.
- **Continuously differentiable**: A good activation function must be continuously differentiable. This enables a smooth learning process while training.
- **Finite Range**: An activation function should have a finite range to map input values into, as it helps normalize data for better performance tuning.
- **Monotonic**: An activation function should be monotonic in nature for a given interval of input values. This enables faster convergence during gradient descent, as it then becomes convex in nature.

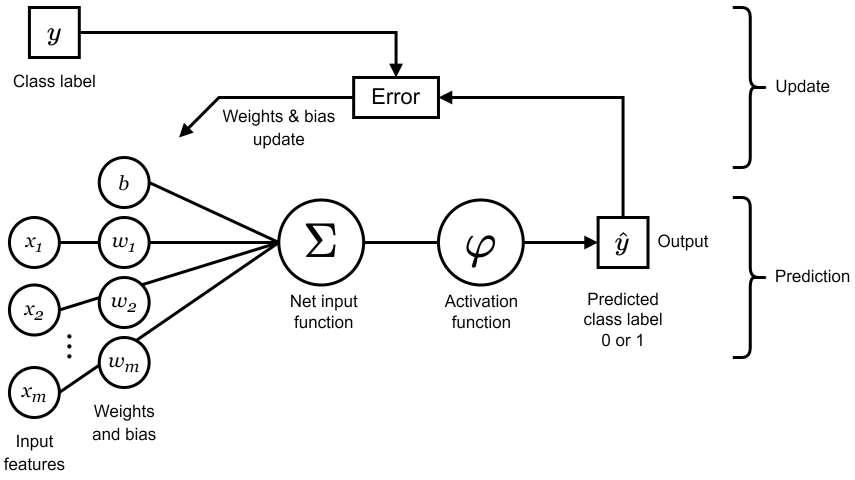


Figure 2: The Perceptron: Example

## 4 Challenges faced by activation functions

Being a hotly-researched topic, there have been a lot of advancements in the domain of finding activation functions, as well as challenges: logical, mathematical and computational. A few such challenges are:

- Vanishing Gradient Problem
- Dead Neuron Problem
- Exploding Gradient Problem
- XOR problem.

### 4.1 Vanishing gradient problem

The Vanishing Gradient is an implementation-based problem in practical scenarios. It occurs during the learning process of neural networks when the product of the parameters and their weights become so low that the system loses track of accuracy, and rounds off the gradient to zero, hence halting the gradient descent, and reaching what is often called a *gradient plateau*.

sigmoid, tanh are some functions that are prone this problem.

### 4.2 Dead neuron problem

The dead neuron problem occurs when the derivative of the last layer is unable to propagate back to the first layer during backpropagation. This is caused due to the same accuracy implementation issues that lead to the Vanishing Gradient problem.

As mentioned above, Sigmoid and tanh are prone to this problem too.

### 4.3 Exploding gradient problem

The Exploding gradient is another implementation-based problem in practical scenarios. It occurs during the learning process of neural networks when the product of a set of weights and inputs becomes so high that it results in an extremely large weight update, often missing the optimal value. We use various mechanisms like dropout to avoid this.

The Binary Step function is prone to this error, as it suddenly jumps from 0 (or -1) to 1 in the neighbourhood of 0.

### 4.4 XOR Problem

Pointed out by *Newell*[3] in 1969, a classic shortcoming of the perceptron has been the XOR problem, one which oscillating activation functions are poised to provide a solution for. The truth table for the XOR function looks like:

| A | B | A XOR B |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

When a single perceptron is paired with one of the popular activation functions, it is unable to find a hyperplane to separate the observations. For example, using the Sigmoid function, we have:

$$W = \sum_{n=0}^{2} w_i.x_i \qquad \rightarrow (1)$$

$$\sigma(W) = \frac{1}{1+e^{-W}} \qquad \rightarrow (2)$$

Here, $w_1$ and $w_2$ are to be learnt and adjusted. We can infer the following from (1):

$$\sigma(x) \geq 0.5$$

if

$$W \geq 0 \implies w_1.x_1 + w_2.x_2 \geq 0$$

Also, when inputs are (1,1), (1, 0) and (0,1), the output should be 0, 1 and 1 respectively.

Therefore,

$$1.w_1 + 1.w_2 < 0 \rightarrow (4.4a)$$
$$1.w_1 + 0.w_2 > 0 \rightarrow (4.4b)$$
$$0.w_1 + 1.w_2 > 0 \rightarrow (4.4c)$$
$$0.w_1 + 0.w_2 < 0 \rightarrow (4.4d)$$

which is impossible. Hence, the popular activation functions cannot solve the XOR problem; we commonly use Multilayer Perceptrons (abbreviated as MLPs) to address it. Until recently, almost all activation functions suffered from this problem.

## 5 Different types of activation functions

Activation functions can be broadly divided into the following categories:

- Linear Activation Function: As their name suggests, the output from these functions is dependent on a linear power of the input. These functions include:
  - Binary Step Function
  - ReLU function (which is a piecewise linear function), and some of its derivatives (LReLU, etc)
- Non-linear Activation Functions: The output of these functions are not confined to a linear power of the inputs. As such, these functions cannot be represented as a straight line. Most activation functions are non-linear in nature. Examples are:
  - Sigmoid
  - Softmax
  - Softplus
  - Hyperbolic tangent
  - ELu

- Swish
- Mish
- GCU

We discuss some of the notable functions below.

### 5.1 Sigmoid

The Binary function proposed above has a lot of flaws that were becoming evident as research in artificial neural networks progressed- it was discreet and not differentiable, resulting in neural networks being limited, and rendering them unable to take advantage of the backpropagation[6] mechanism. *Hinton, Rumelheart et al.* proposed the Sigmoid Function

$$\sigma(x) : \mathbb{R} \to (0, 1)$$

The Sigmoid Function is mathematically defined as:

$$\sigma(x) = \frac{1}{1 + e^{-k.x}}$$

, where $k$ is an arbitrary constant, usually assumed to be 1. It offers a number of advantages over the Binary function:

- It is differentiable.
- The output range $E$ of the function $\epsilon$ $[0, 1]$, which is a continuous range.
- It saturates over a continuous range, making it suitable for backpropagation, for faster and better convergence.
- Calculation of derivative is easy for a sigmoid function. $f\prime(x) = f(x)(1 - f(x))$, so caching can be implemented easily and efficiently in real-life applications.

### 5.2 Tanh

Introduced by *LeCun, Bengio, Hinton, et al.,*[8], the hyperbolic tangent function, represented as $tanh(x)$ mathematically, is a better alternative to the sigmoid function. Tt speeds up the process of convergence by expanding the output range $E$ from $[0, 1]$ to $[-1, 1]$, thus increasing the gradient for the objective function. Also, the average of all the values is 0 (unlike 0.5 in case of sigmoid), which leads to better convergence. The Hyperbolic tangent function is defined as:

$$tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

This function is more resistant towards the problem of Vanishing Gradient, compared to its predecessor Sigmoid.

### 5.3 ReLU

Proposed in 2018 by *Agarap et al.,*[9], the Rectified Linear Unit (ReLU) function is defined as:

$$ReLU(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$$

The ReLU makes it for faster convergence in gradient descent, given the nature of the function (i.e., degree of the polynomial), avoiding saturation and Vanishing Gradient at extreme values (ie., values much greater than 1). ReLU, being a simple $max(0, x)$ function, is also highly efficient computationally, resulting in it being the go-to activation function for many purposes.

### 5.4 LReLU

Leaky ReLU is another function that also takes care of the other extreme values, preventing the Vanishing Gradient Problem at extreme negative values. Leaky ReLU is mathematically defined as:

$$LReLU(x) = \begin{cases} \frac{x}{\alpha} & \text{if } x < 0 \\ x & \text{if } 0 \leq x \end{cases}$$

where:
$\alpha$ is a constant, and $\alpha >> x$.

### 5.5 PReLU

*He, Zhang et al.,*[10] proposed the Parametric ReLU (abbreviated as PReLU) activation function in their paper on ImageNet Classification. Unlike ReLU that zeroes negative inputs, or Leaky ReLU that sets a specific constant $\alpha$ to divide (or conversely multiply) the negative inputs with, PReLU uses a learnable parameter $\beta$, leading to adaptive constants that increase accuracy.

Mathematically, the PReLU activation function is defined as:

$$PReLU(x_i) = \begin{cases} \beta_i & \text{if } x_i < 0 \\ x_i & \text{if } 0 \leq x_i \end{cases}$$

where $\beta$ is a learnable parameter.

## 6 Alternative solutions for challenges faced by activation functions

In recent times, self-gated, non-monotonic activation functions are a hot take among researchers. This is because their non-monotonicity ensures atleast one minimum, thus ensuring a guaratee that a gradient descent will converge. Also, being self-gated means that they can regulate their own values and thresholds (derived from the working mechanism of LSTMs). Some notable new activation functions are:

- Swish
- Mish
- GCU and other oscillating functions

In the forthcoming sections, we discuss these alternative solutions in detail.

### 6.1 Swish

Proposed by *Ramachandran et al.*[11], Swish is one of the latest activation functions. It makes for efficient prevention of both the Exploding and Vanishing Gradient Problems. The Swish activation function is mathematically defined as:

$$Swish(x) = x.\sigma(\beta.x)$$

where $\sigma(x)$ is the Sigmoid function, and $\beta$ is a trainable parameter.

The advantage of the Swish function is that when applied on large datasets and architectures that have very sensitive parameters, it outperforms the other activation functions in most (if not all cases) for deep neural networks, albeit at a price of higher computing power.

### 6.2 Mish

Proposed by *Mishra et al.,*[12], Mish is another one of the latest activation functions out there. It handles

$$Mish(x) = x.tanh(\zeta(x))$$

where $\zeta(x)$ is the Softplus function, defined as:

$$\zeta(x) = ln(1 + e^x)$$

The Mish function outperforms even the Swish function on almost all occassions. This can be intuitively deduced from the fact that the Hyperbolic Tangent function $tanh(x)$ function used by Mish function is inherently better than the Sigmoid function $\sigma(x)$ used by the Swish function. The tradeoff is an even higher computing power than Swish function.

### 6.3 GCU

*Noel, Trivedi, Dutta,. et al.*[13] described the Growing Cosine Unit activation function in their paper. The Growing Cosine Unit (abbreviated as GCU) is a simple oscillating activation function whose amplitude increases with increasing values. Mathematically, the Growing Cosine Unit is defined as:

$$GCU(x) = x.cos(x)$$

#### 6.3.1 GCU for the XOR Problem

Given the mathematical definition of GCU, we consider 0 as -1 for the sake of simplicity. Therefore, the triads are:

$$(\begin{bmatrix}-1\\-1\end{bmatrix}, -1), (\begin{bmatrix}-1\\1\end{bmatrix}, 1), (\begin{bmatrix}1\\-1\end{bmatrix}, 1), (\begin{bmatrix}1\\1\end{bmatrix}, -1)$$

Let $X = w_1.x_1 + w_2.x_2$

| $x_1$ | $x_1$ | $X$ | $GCU(X)$ |
|---|---|---|---|
| -1 | -1 | $-w_1 - w_2$ | -1 |
| -1 | 1 | $w_2 - w_1$ | 1 |
| 1 | -1 | $w_1 - w_2$ | 1 |
| 1 | 1 | $w_1 + w_2$ | -1 |

Now, using the definition of the activation function $GCU(X) = X.cos(X)$, we have:

$$\begin{aligned} GCU(-w_1 - w_2) &= -1 \quad \rightarrow (1) \\ GCU(w_2 - w_1) &= 1 \quad \rightarrow (2) \\ GCU(w_1 - w_2) &= 1 \quad \rightarrow (3) \\ GCU(w_1 + w_2) &= -1 \quad \rightarrow (4) \end{aligned}$$

On solving, we have:

$$(w_1 + w_2).cos(w_1 + w_2) + (w_1 - w_2).cos(w_1 - w_2) = 0$$

The above equation has multiple solutions in the range $[0, 2\pi]$, of which $(0, 0)$ is a trivial solution.

Thus, the oscillating function is successfully circumventing the limitations on activation functions imposed by the XOR Problem.

## 7 Biologically inspired oscillating activation functions

Other than GCU, there are other biologically inspired functions that *Noel, M. M., Bharadwaj, S., et al.*[4] have introduced in their paper. Some of them are:

- Non-Monotonic Cubic Unit (NCU): $f(z) = z - z^3$
- Shifted Quadratic Unit (SQU): $f(z) = z^2 + z$

- Decaying Sine Unit (DSU): $f(z) = \frac{\pi}{2}(\text{sinc}\,(z-\pi) - \text{sinc}\,(z+\pi))$
- Shifted Sinc Unit (SSU): $f(z) = \pi\ \text{sinc}(z - \pi)$

where

$$sinc(x) = \begin{cases} 1 & \text{if } x = 0 \\ \frac{sin(x)}{x} & \text{if } x \neq 0 \end{cases}$$

## 8 Conclusion

Through this paper, the popular and emerging activation functions have been summarized and compared. We have also taken a look at the problems faced by current activation functions (most notably the XOR Problem) and how oscillating activation functions help to overcome them. In a nutshell, this paper showcases how useful oscillating functions can be in a variety of use-cases and scenarios.

## References


[1] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[2] Jan Mycielski. Marvin minsky and seymour papert, perceptrons, an introduction to computational geometry. *Bulletin of the American Mathematical Society*, 78(1):12–15, 1972.

[3] Allen Newell. A step toward the understanding of information processes: Perceptrons. an introduction to computational geometry. marvin minsky and seymour papert. mit press, cambridge, mass., 1969. vi+ 258 pp., illus. cloth, 12;paper, 4.95. *Science*, 165(3895):780–782, 1969.

[4] Matthew Mithra Noel, Shubham Bharadwaj, Venkataraman Muthiah-Nakarajan, Praneet Dutta, and Geraldine Bessie Amali. Biologically inspired oscillating activation functions can bridge the performance gap between biological and artificial neurons. *arXiv preprint arXiv:2111.04020*, 2021.

[5] Kotaro Hirasawa, Masanao Ohbayashi, and Masaru Koga. Forward propagation universal learning network. 九州大學工學部紀要, 55(3):225–234, 1995.

[6] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *nature*, 323(6088):533–536, 1986.

[7] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[8] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[9] Abien Fred Agarap. Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[11] Prajit Ramachandran, Barret Zoph, and Quoc V Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[12] Diganta Misra. Mish: A self regularized non-monotonic activation function. *arXiv preprint arXiv:1908.08681*, 2019.

[13] Mathew Mithra Noel, Advait Trivedi, Praneet Dutta, et al. Growing cosine unit: A novel oscillatory activation function that can speedup training and reduce parameters in convolutional neural networks. *arXiv preprint arXiv:2108.12943*, 2021.